

# Rotor: Bringing .NET to FreeBSD

Stephen R. Walli  
Program Manager, Rotor Project

# Outline

- Project Goals
- The License
- The ECMA standards work
- Rotor architecture
- Platform adaptation layer
- Base Class Library Selection

# What is Rotor?

- Rotor is a reference implementation of the ECMA CLI and C# specs, plus key SDK tools, delivered as source under a license with generous terms for non-commercial purposes.
- It will build and run on Windows and FreeBSD

# Why Are We Building Rotor?

- Key reason: support of proposed ECMA standard
  - Proof point for alternate platforms
  - Approachable “guide” for teachers, experimenters, and other commercial implementers
- ECMA standard relates compact framework, .NET framework, and third-party implementations
- Why bootstrap alternate CLI implementations?
  - Establish new runtime platform around web services
  - Interop will be a key to customer satisfaction
  - “Interop” is not “cross platform” (WORA is non-goal)

# Shared Source CLI License

- Part of Microsoft's shared source framework
  - Multiple licenses offer flexibility and choice
  - Maintains the intellectual property rights needed to support strong software ecosystem
  - Adopts beneficial aspects of open source models, especially in the areas of source transparency and community
- One representative license will be for the shared source CLI, C#, and ECMAScript implementations:
  - Neither published nor final, but similar to published WinCE license
  - Feedback welcome!
- Goals of license design
  - Long-term investment of intellectual property into intellectual commons
  - Very liberal rights w.r.t. non-commercial derivatives, to increase educational access to seed the future
  - "Taintless" viewing of source when used as an implementation guide

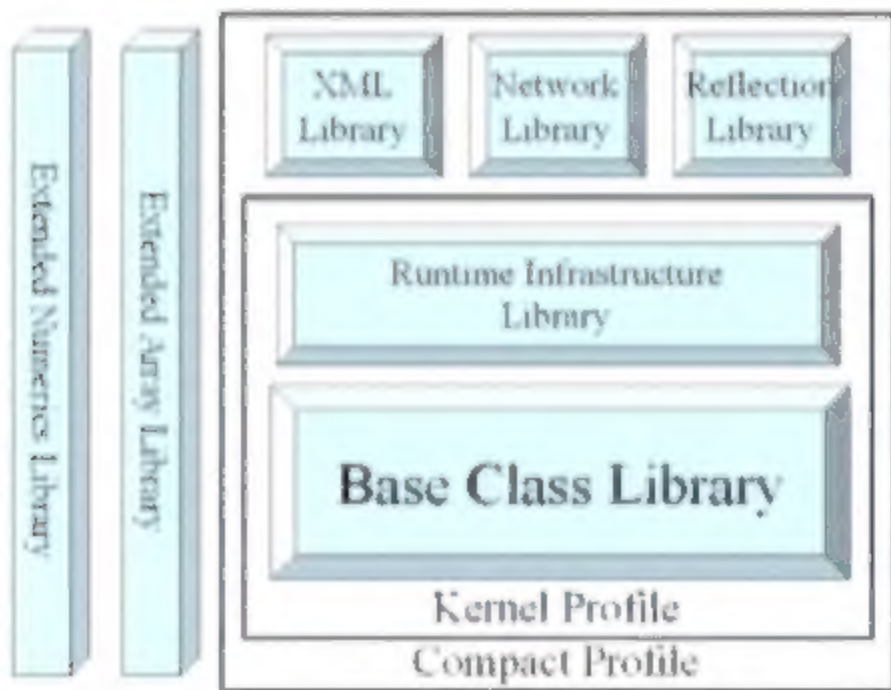
# ECMA Submissions

- **Partition I: Concepts and Architecture**
  - Includes common type system
- **Partition II: Metadata Definition and Semantics**
  - Includes ilasm syntax, file format, PE and metadata
- **Partition III: CIL Instruction Set**
  - Includes instruction validation and verification rules
- **Partition IV: Profiles and Libraries**
  - Describes partitioning and common material
- **Partition V: Annexes**
  - Validation, verification, design guidelines, etc.
- **C# Programming Language**
  - Complete language specification, CLI is runtime for C#

# Detail: Base Class Libraries

- **Abstract OS Interface**
  - Networking, Security, Standard I/O, Threading
- **Common Programming Library**
  - Common Data Types, Advanced Data Types, Collections, Extended Numerics, Regular Expressions, Serialization
- **High-Level Programming**
  - Asynchronous Programming, Globalization, Remoting, XML, Advanced XML
- **EE Functionality**
  - GC, Hosting, NonCLS, Reflection, Unmanaged
- **Miscellaneous**
  - Kernel, Basic Language Support, Development Time

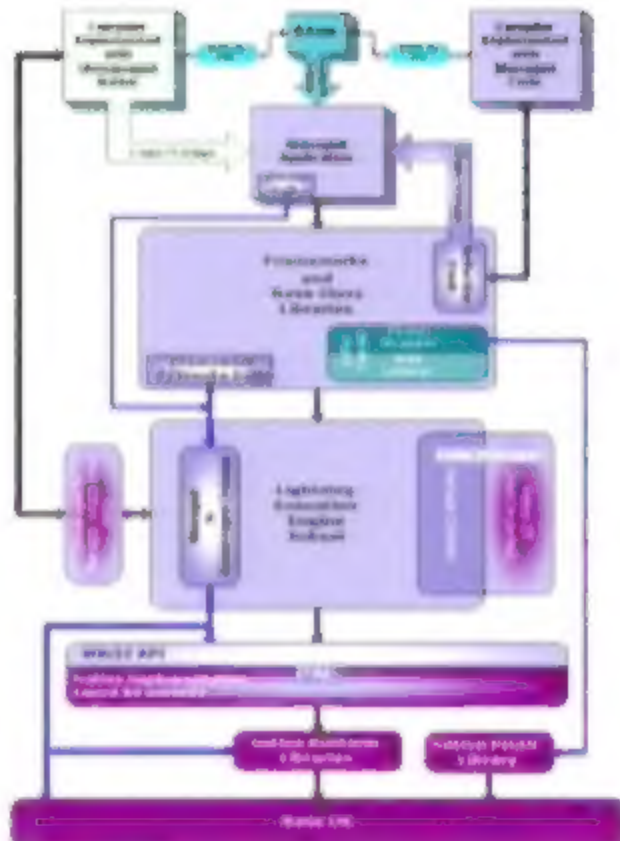
# Map of Libraries within Profiles





## Interesting Rotor Components

- PAL
- Alternate JIT
- Simpler GC
- Hosting and calls into managed code
- Platform Independent GUI



Microsoft Confidential

# PAL

- PAL implements platform support in a generic way
  - Boot loader
  - Threading support
  - Networking
  - Timers
  - File system
- PAL APIs preserve Win32 aspect but are chosen to facilitate porting to Unix platforms
  - Designed to minimize changes in DNoW code base
- PAL used by the
  - CLR
  - FX classes via P/Invoke
  - Unmanaged tools
- About 200 Win32 APIs will be implemented
  - Eliminating WinForms simplified it considerably

# PIGUI Plan

- Many competing desktop environments
  - Unlike the Ecosystem (Ecos)
  - But none take the standard that they want
  - How to go on into the desktop world
- Many desktops exist as different flavors
  - Different desktops
    - GNOME, KDE, Xfce, LXDE, etc.
    - GTK+ vs KDE on top of Qt
    - Many others out there
  - A desktop is a collection of components and libraries
    - X11, Xorg, etc.
  - It has licensing issues on Windows
  - GTK+ has ugly programming model and no API/2 option
  - It is not, not look and feel and possibly slow
- Current plan is to step back
  - Fits with ubiquity goal
  - All the selected system, no option sample

# Rules for Cutting

- Started with everything in DNoW and then cut
  - Didn't build up from ECMA standards
- Cut criteria:
  - Remove trade secret or other sensitive IP
    - In most cases there are alternate low IP code bases
  - Remove Windows-specific functionality that can't feasibly be ported in v1
  - Remove features that are beyond the basics

# Significant Cuts and Exceptions

- **COM interop cut**
  - Most common in target platforms
  - Expensive cut since it usually requires recompiling
- **Windows Forms cut**
  - Too dependent on Win32 to become a cross-platform independent
- **ASP.NET cut**
  - But System Web Services in .NET
- **MS-specific atomic data classes cut**
  - C# server and C++ B providers are not
  - Other means for interop and publication
- **VB.NET support cut**
  - No cross-dev scenario
  - All Designer classes in VB and C# compiler will not be ported
  - But # and J output will be ported
- **Transparent loading and JW support**
  - Native will not load the images
  - Users must use Icon and Image managers

# Availability

- Q1/Q2 2002 for beta release
- Q3/Q4 2002 for final release
- Community build exercise around the release

# Additional Resources

- Project Web site:  
<http://rotor>
- ECMA specs:  
<http://msdn.microsoft.com/net/ecma>
- Whitepaper and FAQ:  
<http://www.microsoft.com/net/whitepapers.asp>
- Microsoft commercial C# and CLR beta:  
<http://msdn.microsoft.com/downloads>
- Shared source info  
<http://www.microsoft.com/sharedsource>  
<http://www.microsoft.com/windows/embedded/ce>

Questions?



## Leader III

### Software on Demand

Hoi Vo

Figure 1. The effect of the concentration of the inhibitor on the rate of polymerization of methyl methacrylate in benzene at 60°C. The concentration of the initiator was 0.001 mole/l. and the concentration of the monomer was 0.5 mole/l. The concentration of the inhibitor was 0.001 mole/l. (○), 0.002 mole/l. (●), 0.004 mole/l. (▲), 0.008 mole/l. (△), 0.016 mole/l. (□), 0.032 mole/l. (◇), 0.064 mole/l. (◇), 0.128 mole/l. (◇), 0.256 mole/l. (◇), 0.512 mole/l. (◇), 1.024 mole/l. (◇), 2.048 mole/l. (◇), 4.096 mole/l. (◇), 8.192 mole/l. (◇), 16.384 mole/l. (◇), 32.768 mole/l. (◇), 65.536 mole/l. (◇), 131.072 mole/l. (◇), 262.144 mole/l. (◇), 524.288 mole/l. (◇), 1048.576 mole/l. (◇), 2097.152 mole/l. (◇), 4194.304 mole/l. (◇), 8388.608 mole/l. (◇), 16777.216 mole/l. (◇), 33554.432 mole/l. (◇), 67108.864 mole/l. (◇), 134217.728 mole/l. (◇), 268435.456 mole/l. (◇), 536870.912 mole/l. (◇), 1073741.824 mole/l. (◇), 2147483.648 mole/l. (◇), 4294967.296 mole/l. (◇), 8589934.592 mole/l. (◇), 17179869.184 mole/l. (◇), 34359738.368 mole/l. (◇), 68719476.736 mole/l. (◇), 137438953.472 mole/l. (◇), 274877906.944 mole/l. (◇), 549755813.888 mole/l. (◇), 1099511627.776 mole/l. (◇), 2199023255.552 mole/l. (◇), 4398046511.104 mole/l. (◇), 8796093022.208 mole/l. (◇), 17592186044.416 mole/l. (◇), 35184372088.832 mole/l. (◇), 70368744177.664 mole/l. (◇), 140737488355.328 mole/l. (◇), 281474976710.656 mole/l. (◇), 562949953421.312 mole/l. (◇), 1125899906842.624 mole/l. (◇), 2251799813685.248 mole/l. (◇), 4503599627370.496 mole/l. (◇), 9007199254740.992 mole/l. (◇), 18014398509481.984 mole/l. (◇), 36028797018963.968 mole/l. (◇), 72057594037927.936 mole/l. (◇), 144115188075855.872 mole/l. (◇), 288230376151711.744 mole/l. (◇), 576460752303423.488 mole/l. (◇), 1152921504606846.976 mole/l. (◇), 2305843009213693.952 mole/l. (◇), 4611686018427387.904 mole/l. (◇), 9223372036854775.808 mole/l. (◇), 18446744073709551.616 mole/l. (◇), 36893488147419103.232 mole/l. (◇), 73786976294838206.464 mole/l. (◇), 147573952589676412.928 mole/l. (◇), 295147905179352825.856 mole/l. (◇), 590295810358705651.712 mole/l. (◇), 1180591620717411303.424 mole/l. (◇), 2361183241434822606.848 mole/l. (◇), 4722366482869645213.696 mole/l. (◇), 9444732965739290427.392 mole/l. (◇), 18889465931478580854.784 mole/l. (◇), 37778931862957161709.568 mole/l. (◇), 75557863725914323419.136 mole/l. (◇), 151115727451828646838.272 mole/l. (◇), 302231454903657293676.544 mole/l. (◇), 604462909807314587353.088 mole/l. (◇), 1208925819614629174706.176 mole/l. (◇), 2417851639229258349412.352 mole/l. (◇), 4835703278458516698824.704 mole/l. (◇), 9671406556917033397649.408 mole/l. (◇), 19342813113834066795298.816 mole/l. (◇), 38685626227668133590597.632 mole/l. (◇), 77371252455336267181195.264 mole/l. (◇), 154742504910672534362390.528 mole/l. (◇), 309485009821345068724781.056 mole/l. (◇), 618970019642690137449562.112 mole/l. (◇), 1237940039285380274899124.224 mole/l. (◇), 2475880078570760549798248.448 mole/l. (◇), 4951760157141521099596496.896 mole/l. (◇), 9903520314283042199192993.792 mole/l. (◇), 19807040628566084398385987.584 mole/l. (◇), 39614081257132168796771975.168 mole/l. (◇), 79228162514264337593543950.336 mole/l. (◇), 158456325028528675187087900.672 mole/l. (◇), 316912650057057350374175801.344 mole/l. (◇), 633825300114114700748351602.688 mole/l. (◇), 1267650600228229401496703205.376 mole/l. (◇), 2535301200456458802993406410.752 mole/l. (◇), 5070602400912917605986812821.504 mole/l. (◇), 10141204801825835211973625643.008 mole/l. (◇), 20282409603651670423947251286.016 mole/l. (◇), 40564819207303340847894502572.032 mole/l. (◇), 81129638414606681695789005144.064 mole/l. (◇), 162259276829213363391578010288.128 mole/l. (◇), 324518553658426726783156020576.256 mole/l. (◇), 649037107316853453566312041152.512 mole/l. (◇), 1298074214633706907132624082305.024 mole/l. (◇), 2596148429267413814265248164610.048 mole/l. (◇), 5192296858534827628530496329220.096 mole/l. (◇), 10384593717069655257060992658440.192 mole/l. (◇), 20769187434139310514121985316880.384 mole/l. (◇), 41538374868278621028243970633760.768 mole/l. (◇), 83076749736557242056487941267521.536 mole/l. (◇), 166153499473114484112975882535043.072 mole/l. (◇), 332306998946228968225951765070086.144 mole/l. (◇), 664613997892457936451903530140172.288 mole/l. (◇), 1329227995784915872903807060280344.576 mole/l. (◇), 2658455991569831745807614120560689.152 mole/l. (◇), 5316911983139663491615228241121378.304 mole/l. (◇), 10633823966279326983230456482242756.608 mole/l. (◇), 21267647932558653966460912964485513.216 mole/l. (◇), 42535295865117307932921825928971026.432 mole/l. (◇), 85070591730234615865843651857942052.864 mole/l. (◇), 170141183460469231731687303715884105.728 mole/l. (◇), 340282366920938463463374607431768211.456 mole/l. (◇), 680564733841876926926749214863536422.912 mole/l. (◇), 1361129467683753853853498429727072845.824 mole/l. (◇), 2722258935367507707706996859454145691.648 mole/l. (◇), 5444517870735015415413993718908291383.296 mole/l. (◇), 10889035741470030830827987437816582766.592 mole/l. (◇), 21778071482940061661655974875633165533.184 mole/l. (◇), 43556142965880123323311949751266331066.368 mole/l. (◇), 87112285931760246646623899502532662132.736 mole/l. (◇), 174224571863520493293247799005065

# Agenda

1. Einführung

2. Grundlagen der Informatik

3. Algorithmen und Datenstrukturen

4. Programmierung

5. Systemarchitektur

# Introduction



MoneyCentral

Home About Us Contact Us Privacy Policy Terms of Service

Home > About Us > Contact Us > Privacy Policy > Terms of Service

Home > About Us > Contact Us

Privacy Policy

Home > About Us > Contact Us > Privacy Policy

Home > About Us > Contact Us

# Real challenges

1. *How can we make the most of the data we have?*

2. *How can we make the most of the data we don't have?*

3. *How can we make the most of the data we don't want?*

4. *How can we make the most of the data we don't know we have?*

5. *How can we make the most of the data we don't know we don't have?*

6. *How can we make the most of the data we don't know we don't know we have?*

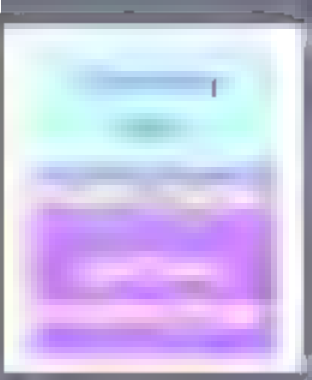
7. *How can we make the most of the data we don't know we don't know we don't have?*

8. *How can we make the most of the data we don't know we don't know we don't know we have?*

9. *How can we make the most of the data we don't know we don't know we don't know we don't have?*

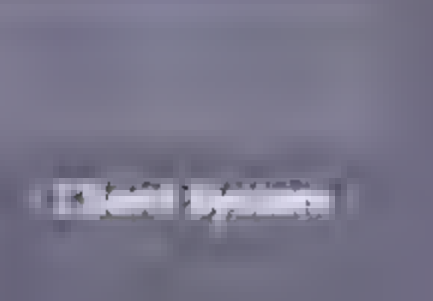
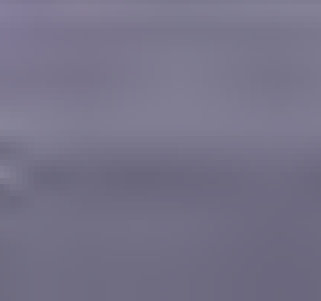
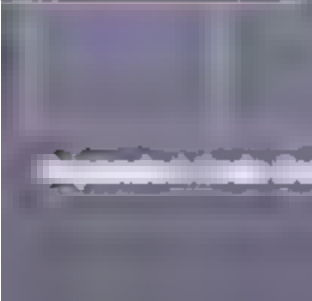
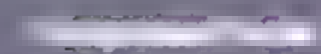
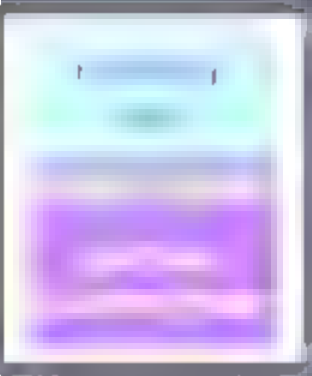
10. *How can we make the most of the data we don't know we don't know we don't know we don't know we have?*

# Loader III

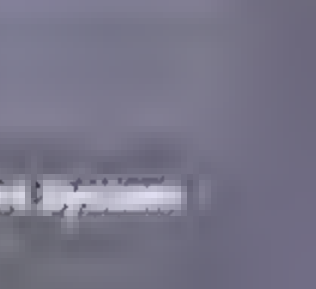
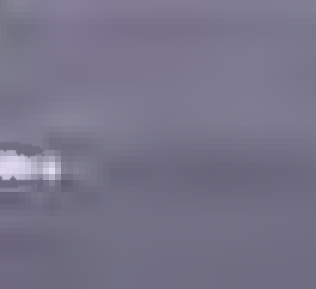
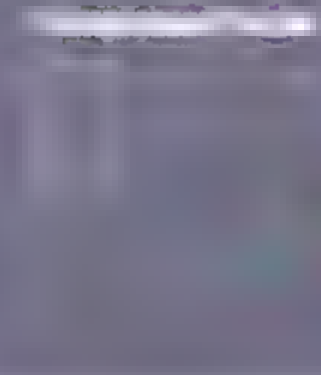
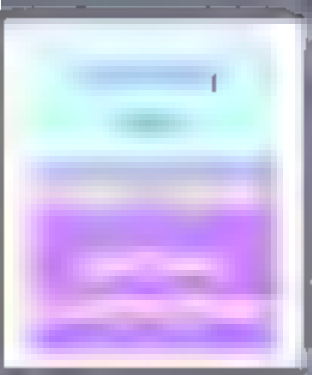


1. 100 2. 100 3. 100 4. 100 5. 100 6. 100 7. 100 8. 100 9. 100 10. 100

# Board 11



# Loadcell



# Technical Challenges

• **Complexity**

• **Integration**

• **Scalability**

• **Performance**

• **Security**

• **Interoperability**

• **Flexibility**

• **Reliability**

• **Cost**

• **Time to Market**

• **Support**





# Technical Challenges

• **Complexity**  
• **Scalability**

• **Security**

• **Performance**

• **Integration** | **Interoperability**

• **Compliance** | **Regulatory Requirements**

• **Resource Management**

• **Documentation**

• **Testing**

• **Deployment**



# Sparse Binary Connet

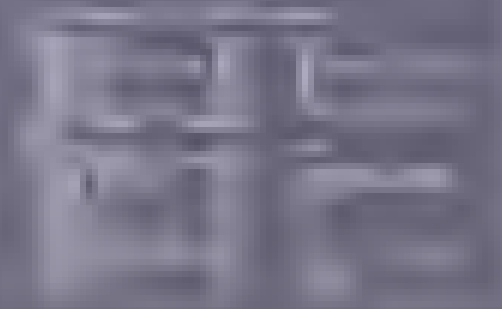


# Sparse Binary Compression

problem

Given a binary matrix  $M$  of size  $n \times m$ ,  
find a binary matrix  $\tilde{M}$  of size  $n \times m$  such that  
 $\tilde{M}$  is a sparse binary matrix (i.e.  $\tilde{M}$  has  
at most  $k$  non-zero entries per row)  
and  $\tilde{M}$  is close to  $M$  (i.e.  $\|M - \tilde{M}\|_F$  is small)

compression ratio



# Sparse Binary Coding

problem



compression ratio



# Sparse Binary Concretization

problem

compression ratio

upgrade



## Spare Day / Concert

## problem

compression ratio

upgrade

# Handling Data Blocks

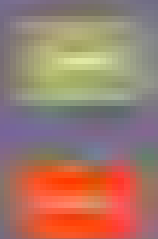
- Issues:

- Replication (redundancy) & storage efficiency & availability
- Data access (storage & retrieval) & security & backup & recovery
- File-level data access & record-level access
- Security of data & its use & its management
- Data performance & storage management

# Handling Data Blocks

## Issues:

- 1. Minimizing redundancy among different data copies
- 2. Replicating blocks across multiple data storage devices
- 3. Replicating data for recovery against corruption
- 4. Replicating the data across different domains
- 5. Replicating data for backup purposes





# Handling Data Blocks

- Issues:

- Multiple references from different code paths
- R/W data blocks require stricter sync during decomp
- Too large data arrays require special handling
- Special PE data blocks cannot be streamed
- Singly referenced data blocks detection



# Handling Data Blocks

- Issues:

- Multiple references from different code paths
- R/W data blocks require stricter sync during decomp
- Too large data arrays require special handling
- Special PE data blocks cannot be streamed
- Singly referenced data blocks detection



# Download Cache Optimization

- Use BBT profile data for static layout
- Cached funclet layout schemes:
  - Layout based on funclet loading order
  - Layout based on BBT layout order
  - Layout based on pre-computed order
- One download cache for each Loader.NET component
- Multiple processes share same download cache for the same component

# Into the Future

- **MSIL streaming support**
  - Metadata per class
  - CLR hook for class loading and release
  - Pre-JIT binary support
- **Integration into platform (i.e. Fusion)**
- **Offline application support**
  - Drizzle mode support
- **Universal download cache**